CLAIMS

1. A computer implemented method of determining, in a computer environment, the equivalence, if any, of two blocks of assignment statements in a computer program, for use in compiler optimisation of source code, program verification, program proving, and like computing tasks, said method comprising the steps of:

(a) forming, for each block of assignment statements, a corresponding array, each array comprising a plurality of elements corresponding to respective ones of the statements and populating the elements with attributes of the statements including the expression at the right-hand side of the statement;

(b) processing, in each array, each assignment statement in turn, in the order from the last statement to the first, the processing comprising the inspection of each unprocessed assignment statement in turn, in the order from the last unprocessed assignment statement to the first, to determine if the variable appearing on the left-hand side of the unprocessed assignment statement appears on the right-hand side of the assignment statement being processed;

(c) during step (b), in each array, if the variable appearing on the left-hand side of the unprocessed assignment statement also appears on the right-hand side of the assignment statement being processed, replacing all occurrences of such variable on the right-hand side of the assignment statement being processed, non-recursively, by the right-hand side of the said unprocessed assignment statement;

(d) forming, from each array, a corresponding new block of assignment statements comprising the statements processed according to steps (b) and (c) less any statements which, after processing, is either an identity (the left and right sides of the statement are identical) or whose left-hand side variable is not one of the output variables;

(e) creating, from each new block of assignment statements, a corresponding new array, each array comprising a plurality of elements corresponding to respective ones of the statements and populating the elements with attributes of the statements including the expression at the right-hand side of the statement;

(f) sorting, in each new array, the array elements in alphabetical order using the output variable name as the key.

(g) comparing the arrays to detect the equivalence of two blocks of assignment statements.

2. A method according to Claim 1, including, for each assignment statement in a block, testing the statement for compliance with predetermined rules concerning the applicability of steps (b) and (c), and, if said rules are not complied with, abandoning the method with an error message.

3. A method according to Claim l, whereby Step (a) is preceded by a formatting step of the right-hand side of each assignment statement according to predetermined rules.

4. A method according to Claim 1 whereby in Step (d) the right-hand side of each included assignment statement is formatted according to predetermined rules.

5. A method according to Claim 1, whereby at the conclusion of Step (e) if the number of assignment statements is not equal to the number of output variables, abandoning the method with an error message.

6. An apparatus to determine, in a computer environment, the equivalence, if any, of two blocks of assignment statements in a computer program, for use in compiler optimisation of source code, program verification, program proving, and like computing tasks, said apparatus comprising :

(a) forming means for forming, for each block of assignment statements, a corresponding array, each array comprising a plurality of elements corresponding to respective ones of the statements and populating the elements with attributes of the statements including the expression at the right-hand side of the statement;

(b) processing means for processing, in each array, each assignment statement in turn, in the order from the last statement to the first, the processing comprising the inspection of each unprocessed assignment statement in turn, in the order from the last unprocessed assignment statement to the first, to determine if the variable appearing on the left-hand side of the unprocessed assignment statement appears on the righthand side of the assignment statement being processed;

(c) during step (b), in each array, if the variable appearing on the left-hand side of the unprocessed assignment statement also appears on the right-hand side of the assignment statement being processed, replacement means for replacing all occurrences of such variable on the right-hand side of the assignment statement being processed, non-recursively, by the right-hand side of the said unprocessed assignment statement;

(d) forming means for forming, from each array, a corresponding new block of assignment statements comprising the statements processed according to steps (b) and (c) less any statements which, after processing, is either an identity (the left and right sides of the statement are identical) or whose left-hand side variable is not one of the output variables;

(e) creation means for creating, from each new block of assignment statements, a corresponding new array, each array comprising a plurality of elements corresponding to respective ones of the statements and populating the elements with attributes of the statements including the expression at the right-hand side of the statement;

(f) sorting means for sorting, in each new array, the array elements in alphabetical order using the output variable name as the key.

(g) comparison means for comparing the arrays to detect the equivalence of two blocks of assignment statements.

7. A computer program product including a computer readable medium having recorded thereon a computer program for determining, in a computer environment, the equivalence, if any, of two blocks of assignment statements in a computer program, for use in compiler optimisation of source code, program verification, program proving, and like computing tasks, said program comprising:

(a) forming process steps for forming, for each block of assignment statements, a corresponding array, each array comprising a plurality of elements corresponding to respective ones of the statements and populating the elements with attributes of the statements including the expression at the right-hand side of the statement;

(b) processing steps for processing, in each array, each assignment statement in turn, in the order from the last statement to the first, the processing comprising the inspection of each unprocessed assignment statement in turn, in the order from the last unprocessed assignment statement to the first, to determine if the variable appearing on the left-hand side of the unprocessed assignment statement appears on the right-hand side of the assignment statement being processed;

(c) during step (b), in each array, if the variable appearing on the left-hand side of the unprocessed assignment statement also appears on the right-hand side of the assignment statement being processed, replacement steps for replacing all occurrences of such variable on the right-hand side of the assignment statement being processed, non-recursively, by the right-hand side of the said unprocessed assignment statement;

(d) forming process steps for forming, from each array, a corresponding new block of assignment statements comprising the statements processed according to steps (b) and (c) less any statements which, after processing, is either an identity (the left and right sides of the statement are identical) or whose left-hand side variable is not one of

the output variables;

(e) creation process steps for creating, from each new block of assignment statements, a corresponding new array, each array comprising a plurality of elements corresponding to respective ones of the statements and populating the elements with attributes of the statements including the expression at the right-hand side of the statement;

(f) sorting process steps for sorting, in each new array, the array elements in alphabetical order using the output variable name as the key.

(g) comparison process steps for comparing the arrays to detect the equivalence of two blocks of assignment statements.